SRP: The Single Responsibility Principle

*A CLASS SHOULD HAVE ONLY ONE REASON TO CHANGE.*

Thinking about in a first looks this principle looks like only one functionality for one class. In other way there will be only one method. But **that’s not true!** There is a difference between functionality and reason of to create a class. If a class has more than one reason to change, it has more than one responsibility. Classes with more than a single responsibility should be broken down into smaller classes, each of which should have only one responsibility and reason to change.

Let’s take a simple example by looking out the below implementation of class

**What is a Responsibility?**

In the context of the Single Responsibility Principle (SRP) we define a responsibility to be

“a reason for change.” If you can think of more than one motive for changing a class, then

that class has more than one responsibility. This is sometimes hard to see.

For example, consider the Modem interface

in Listing 8-1. Most of us will agree that this interface looks perfectly reasonable. The four

functions it declares are certainly functions belonging to a modem.

**Conclusion**

The SRP is one of the simplest of the principles, and one of the hardest to get right.

Joining responsibilities is something that we do naturally. Finding and separating those

responsibilities from one another is much of what software design is really about. Indeed,

the rest of the principles we will discuss come back to this issue in one way or another.